DICTATE: DIstributed CerTification Authority with probabilisTic frEshness for Ad Hoc Networks

Jun Luo, *Student Member*, *IEEE*, Jean-Pierre Hubaux, *Senior Member*, *IEEE*, and Patrick T. Eugster

Abstract—Securing ad hoc networks is notoriously challenging, notably due to the lack of an online infrastructure. In particular, key management is a problem that has been addressed by many researchers but with limited results. In this paper, we consider the case where an ad hoc network is under the responsibility of a *mother certification authority* (mCA). Since the nodes can frequently be collectively isolated from the mCA (e.g., for a remote mission) but still need the access to a certification authority, the mCA preassigns a special role to several nodes (called *servers*) that constitute a *distributed certification authority* (dCA) during the isolated period. We propose a solution, called DICTATE (DIstributed CerTification Authority with probabilisTic frEshness), to manage the dCA. This solution ensures that the dCA always processes a certificate update (or query) request in a finite amount of time and that an adversary cannot forge a certificate. Moreover, it guarantees that the dCA responds to a query request with the most recent version of the queried certificate in a certain probability; this probability can be made arbitrarily close to 1, but at the expense of higher overhead. Our contribution is twofold: 1) a set of certificate management protocols that allow trading protocol overhead for certificate freshness or the other way around, and 2) a combination of threshold and identity-based cryptosystems to guarantee the security, availability, and scalability of the certification function. We describe DICTATE in detail and, by security analysis and simulations, we show that it is robust against various attacks.

Index Terms—Ad hoc networks, system design, security, public-key infrastructure, Quorum Systems, simulations.

1 Introduction

Ad hoc networks are collections of peered mobile nodes that communicate through wireless links. Such networks require stringent security protocols to protect their nodes from various attacks [1], [2]. However, implementing these protocols is difficult because these networks are constructed without using an online infrastructure and because the wireless links are particularly vulnerable. In this paper, we design a secure and efficient *public-key infrastructure* (PKI) for ad hoc networks.

Public-key cryptography supports mechanisms that achieve security objectives such as confidentiality, authentication, and nonrepudiation. It can also pave the way for applying symmetric-key cryptography by bootstrapping a secured channel through mutual authentication and the establishment of a shared secret. However, a carefully planned PKI[1] is necessary to implement these security mechanisms. In the Internet, PKIs (e.g., [3]) usually involve a *certification authority* (CA), which is a *trusted third party* (TTP) that certifies the authenticity of the binding between a public key and its subject entity. Whereas a CA can be implemented in a centralized server for a certain authority domain, a distributed implementation [4] could be preferable for improving availability. As an alternative, PGP [5] is a more flexible PKI that enables users to enjoy public-key cryptography without any support from a CA.

In ad hoc networks, centralized CAs can work only for small authority domains, since the availability of such a CA would be problematic in a large domain due to the highly dynamic network topology. Based on this consideration and on previous results for wired networks, existing proposals for building a PKI in ad hoc networks can be classified into two main trends: 1) A single authority domain across the whole network with a distributed implementation of CA [1], [2] and 2) multiple authority domains of small sizes with a "centralized" authority for each [6].

In addition to distributing the certification authority, applying the joint authority approach [7], [8] can further increase the security of a CA in large distributed systems. This approach advocates the combination of an offline *identification authority* (IA) and an online *revocation authority* (RA). The IA authenticates the **initial** binding between a public key and its subject entity, and the RA keeps track of the status of certificates issued by the IA. Thanks to this separation, compromising the online authority (which is usually more vulnerable than an offline authority) does not enable the adversary to issue certificates to new users, which limits the consequent damages. In spite of the

1. In this paper, the term PKI is used in a broad sense. It refers to public-key management systems offering basic functions of certificate (or key) issuance, validation and revocation.

- *J. Luo and J.-P. Hubaux are with the School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland. E-mail: {jun.luo, jean-pierre.hubaux}@epfl.ch.*
- *P.T. Eugster is with the Department of Computer Sciences, Purdue University, 250 N. University Street, West Lafayette, Indiana, 47907-2066. E-mail: patrick.eugster@cs.purdue.edu.*

Manuscript received 19 Nov. 2004; revised 26 May 2005; accepted 11 Aug. 2005; published online 3 Nov. 2005.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0164-1104.

apparent advantage, no known proposal for ad hoc networks has adopted this approach.

Our proposal improves the joint authority approach to build a PKI for ad hoc networks. Our approach uses an offline IA to issue initial certificates and also to assign special nodes (or servers) to constitute an online **distributed** RA. We propose a DIstributed CerTification Authority with probabilisTic frEshness (DICTATE) to manage the online RA. DICTATE applies threshold cryptography to distribute trust among the RA servers. It also makes use of the services provided by PILOT, a group communication system we proposed in [9]. To issue a (public-key) certificate, DICTATE requires a defined number of RA servers to sign the certificate and then replicates it in a quorum of RA servers. In response to a certificate query, DICTATE forwards it again to a quorum. Our underlying PILOT system guarantees that one quorum forms a probabilistic intersection with another, so that in practice a certificate query acquires, with a high probability, the most recent status of the targeted certificate. Our solution is network friendly: DICTATE can tune the protocol performance **online** to a desired trade-off between the freshness (of a certificate status) and overhead (to achieve the freshness), according to the level of the required freshness and network resource consumption.

The remainder of this paper is structured as follows: Section 2 surveys the existing solutions and summarizes the motivations for our work. Section 3 details the problem to be solved and the system model. Section 4 presents our joint authority design and DICTATE protocols. Section 5 analyzes DICTATE against different attacks. Section 6 simulates DICTATE and compares the results with analytical results. Section 7 concludes the paper.

2 Related Work

This section surveys PKIs in both wired networks and ad hoc networks. It shows that certain principles can be inherited from wired networks for implementing a PKI in ad hoc networks, but a direct migration would not perform well. We further summarize the rationale that leads to the design of our protocols at the end.

2.1 PKIs in Wired Networks

The X.509-based public-key infrastructure [3], a representative of PKIs in wired networks, implicitly assumes centralized CAs. While the availability of such a centralized authority may become a problem in large distributed systems, individual CAs, whose compromise would paralyze the certification function of their domains, also appear to be single points of failure for security. The distributed implementation of a CA (e.g., Ω [4] and COCA [10]) improves the availability of the certification function by organizing different certification servers into a peer-based structure. It also enhances the robustness of the authority against a certain amount of server failures through the use of threshold cryptography. However, all these benefits are obtained at the cost of additional protocol complexity. Particularly, maintaining a reliable group communication system [4] or Byzantine quorum systems [10] is not a trivial task.

PGP [5], an alternative to the PKI based on trusted authorities, provides practical security to protect low-value communications, such as e-mails. PGP is based on referral certification, which allows multiple users to "recommend" a certain user by signing certificates of its public key. This scheme is not perfectly secure because, for example, dishonest users may issue false certificates to cheat other users. The third solution to implement a PKI, SPKI/SDSI [11] has an egalitarian design similar to PGP. It circumvents the dependency on global name spaces, to which both X.509 and PGP are subject, with the concept of *linked local name spaces*.

2.2 PKIs in Ad Hoc Networks

The need for a PKI in an ad hoc network is due to the security requirements of various mechanisms, especially routing (e.g., [12], [13], [14]). However, the distinctive features of ad hoc networks lead to designs of PKIs that are different from those in wired networks.

Zhou and Haas [1] explore the issue of distributed CA in ad hoc networks, with the assumption of a single authority domain across the network. Their solution achieves availability by replicating certificates in multiple servers and employs threshold cryptography to thwart various attacks. However, [1] does not contain a full description of protocols to maintain and control the access to the distributed CA. Luo et al. [2], [15] extend the work of Zhou and Haas by distributing the authority throughout the network. Their proposal focuses on performance: Only localized protocols are used to access the certification function. Unfortunately, the online identification service provided by [2] seems to be vulnerable to the Sybil attack [16], where an attacker can take enough shares within the CA by claiming several identities and can thus reconstruct the system's private key.

Hubaux et al. [17], [6] follow another approach by assuming each node to be its own authority domain. As a counterpart of PGP in ad hoc networks, the self-organized public-key management in [6] allows nodes to certify each other. With the assumption of transitive trust among nodes, appropriate certificate chains are found to verify the certificate of a public key. The disadvantage with such a scheme is that the assumption of a transitive trust could be too strong for mobile networks.

Recently, several proposals have extended the aforementioned approaches in different ways. Extensions to Zhou and Haas's work usually try to solve two problems left open in [1]: 1) how to select the CA servers and 2) how to maintain the CA. Yi and Kravets [18] suggest selectively assigning powerful nodes as CA servers and apply multiple unicastings for accessing CA. They do not explicitly explain how the initial identification is performed when a node joins a network for the first time. Bechler et al. [19] introduce a cluster-based architecture for supporting a distributed CA. However, the online identification service relies on referral certifications from existing network members, which could weaken the security level of the system. Khalili et al. [20] apply an identity-based approach instead of the certificate-based one [18], [19]. They use a set

of preconfigured nodes to form the distributed authority and apply localized protocols similar to those of [2] for the key generation service. Unfortunately, key revocations appear to be difficult because the key generation service refuses to issue keys for a particular identity more than once in order to thwart identity spoofing.

An extension to the trust model in [6] is described in [21]. The trust chain and recommendation protocol used in [6] are again applied but supplemented by a reference protocol. This solution does not commit to perfect security since it is dedicated to low-value communications. A downside of such protocols is that they do not address the security issues with a network-oriented point of view (for example, the topology of a trust graph does not match the changing network topology), which could impair their viability in ad hoc networks. Montenegro and Castelluccia [22] describe a way of binding an identity to its public key without the need of a certificate: The hash of the public key is used as part of the IP address. Unfortunately, a node would have to change its "name" (or identity) upon a key revocation.

A common weakness of PKIs in ad hoc networks is the lack of proper revocation mechanisms. While proposals in [20], [19], [21] do not address the certificate revocation, the solutions in [2], [6], [18] rely on proactive mechanisms to **push** a certificate revocation list (CRL) to other nodes. Although no quantitative evaluation is provided in [2], [18], there is no doubt that proactive pushing, by flooding the network, consumes network resources constantly. Therefore, an on-demand scheme that queries the status of a certificate in question would be more suitable for ad hoc networks.

2.3 Lessons from the Past

According to the experiences from the previous work mentioned above, we summarize our design rationale for the PKI in ad hoc networks as follows:

- Authentication techniques differ in the level of protections that they provide for the targeted communications. Relying on a TTP as an authority yields a high-level protection to secure high-value communications in large-scale networks.
- Performing initial identifications with a full online certification service is questionable. The joint authority approach that integrates an offline IA and an online distributed RA can achieve both security and availability of the CA.
- The proactive share refreshing, which is already expensive enough in wired networks, is not suitable for ad hoc networks (except an extreme case [2] where localized protocols apply). Certain out-of-band mechanisms have to be used for refreshing key shares.
- The network performance should be kept in mind when addressing security issues. Security protection can be somewhat sacrificed to spare network resources in some cases.
- Certificate revocation is an important but often neglected CA service. In wireless networks, it becomes even more significant because (suspected) key compromises can happen more often.

3 Goal and Model

This section states the problem to be solved and models the considered environment.

3.1 Problem Statement

We consider a relatively large-scale ad hoc network (consisting of tens or even hundreds of nodes) with a random mobility pattern, i.e., nodes moving independently within a given field. The network (real life examples include networks temporarily built for rescue or exploration operations) has intermittent connections to a *mother certification authority* (or mCA), which is a trusted authority connected to the backbone with its public data (including its public key) known to all wireless nodes. When the network is disconnected from the mCA (e.g., to perform a rescue operation), it requires a CA that serves requests from nodes to update their public key or to query public key certificates of other nodes. We first give the three properties of a secure CA in such a network as follows:

- **Liveness**: The CA always processes a request in a finite amount of time.
- **Safety**: An adversary is never able to forge a certificate.
- **Freshness**: A query to the CA returns the most recent status of a targeted certificate.

Then, we specify our *CA with probabilistic freshness*: It meets the liveness and safety properties, and it ensures the freshness property with a probability that is termed $\mathcal{F}_d$, or *freshness degree*.

For any correct node, our goals are to:

1. make the CA compliant with the specifications above,
2. provide a certain flexibility for the CA to tune the protocol performance (with respect to $\mathcal{F}_d$ and the overhead) **online** to the desired trade-off,
3. maintain an adequate efficiency, i.e., incur reasonable overhead (defined as the *network load* [9]) even in the case of a high freshness requirement, and
4. keep all protocols transparent to a human user.

3.2 System and Adversary Model

We assume that each node owns a unique identity. We also assume that, in the network, there is a subset (typically 10 to 20 percent of the network)[2] $\mathbf{N}$ ($|\mathbf{N}| = n$) of securely protected and computationally powerful nodes (which the mCA can identify and will use to constitute a distributed CA). A subset $\mathbf{T} \subset \mathbf{N}$ can be compromised during a certain period of time, where $|\mathbf{T}| = t < n/3$.[3] The remaining part of $\mathbf{N}$ consists of a set $\mathbf{C}$ of correct nodes.

2. It is not our goal in this paper to find the optimal size for this set, but we note that generally the larger the size is, the more heavily the network is loaded, whereas the load on individual nodes becomes smaller.

3. Although probabilistic quorum systems [23] (unlike Byzantine quorum systems [24]) still work if $t \geq n/3$ and a threshold cryptosystem requires only $t < n/2$, the system performance degrades dramatically when t goes beyond $n/3$. Therefore, we require $t < n/3$ even though our CA is based on the principle of probabilistic quorum systems.

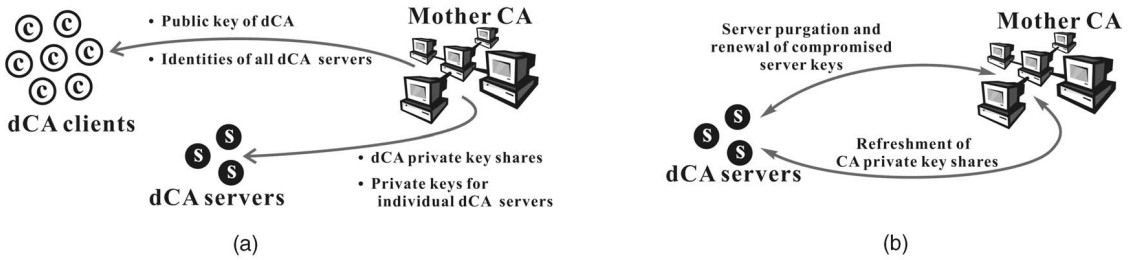


Fig. 1. Operating principles of the joint CA: (a) initialization phase and (b) checktime.

For the adversary model, we consider the following attacks that could be mounted by a malicious node:

- **Impersonation**: A node pretends to be someone else to submit a certificate update.
- **Key Compromise**: A node tries to compromise the private keys of other nodes.
- **Denial of Service (DoS)**: A node tries to slow down the CA by clogging the resources (especially communication resources).
- **Misc**: A node launches eavesdropping, message insertion, corruption, deletion, and replay attacks.

In addition to these attacks, a compromised node may exhibit Byzantine failure, i.e., deviate arbitrarily from the (DICTATE) protocol specification.

4 Our Solution: Joint Authority and DICTATE

Our solution takes the joint authority approach [7], [8]: The mCA (which is connected to the backbone) acts as the offline IA and it assigns the set **N** of special nodes to constitute a distributed CA (dCA hereafter) that performs the role of the online RA (Fig. 1). The dCA nodes are named *server*s, and other nodes are named *client*s. The mCA controls the admission of a node (either a server or a client) to the network at its command, through the issuance of a certificate that asserts the binding between the identity and initial public key of the node. When the network is disconnected from the mCA, clients submit their requests to the dCA. On one hand, a *query* request returns the public key certificate of another client. On the other hand, an *update* request updates a client's public key certificate stored in the dCA. Our solution prevents key compromises through the following revocation[4] mechanism: A certificate should be periodically updated or it will become invalid. The dCA guarantees the legitimacy of an update only if the client who submits the request does not have its private key been stolen or been compromised. Otherwise, the client has to reidentify itself with the mCA through out-of-band mechanisms. A client does not always query a certificate in the case of secure communication; it performs a query only if it suspects the validity of the certificate (e.g., due to a relatively small version number, which we will explain later in this section) that it obtains directly from another client.

Our dCA has a public/private key pair. This public-key pair is based on RSA.[5] The public key, which bears a certificate issued by the mCA, is known to the whole network. The private key of the dCA is shared among the dCA servers by a $(t+1, n)$ robust threshold cryptosystem [27]. A credential generated with the private key testifies the authority of the whole dCA. Each dCA server, like a usual node, has its own public/private key pair. We apply an identity-based scheme [28] and a corresponding signature scheme [29] for this public-key pair and make the identities of all dCA servers publicly known, such that any network node knows the public key of a dCA server. The private key of a server is generated by the mCA; it only represents the authority of an individual server. Applying an identity-based cryptosystem incurs much less communication overhead (we elaborate on this issue when explaining the key revocation in the next paragraph, and we also describe the usage of this cryptosystem in Sections 4.2 and 4.3). The system initialization is illustrated in Fig. 1a.

Periodically,[6] there is a *checktime*, as shown in Fig. 1b, at which the dCA servers (physically) go back to the mCA for a "purgation" (only dCA servers should go through this procedure; clients can still perform their remote operations). During the checktime, the mCA, through out-of-band mechanisms, detects compromised servers and has them reinitiated or substituted by new ones; it also refreshes the secret shared among the dCA. Since online detection of compromised servers is hard and cooperative detection schemes [26] may suffer from blackmail attacks, offline detection seems to be the only way to guarantee security. Note that server identities are fixed, and the key revocation of a server is done by using as public key the combination of the identity and the timestamp corresponding to a certain *checktime interval* [28] (i.e., no information such as revocation lists or certificates needs to be distributed). A client can verify the validity of a message from any server with the knowledge of its identity and a local clock loosely synchronized with the checktime. The above descriptions show that, unlike the traditional joint authority approach, the mCA not only provides identification service but also manages the dCA. Therefore, we use the terms mCA and dCA instead of IA and RA in this paper.

4. Strictly speaking, our solution revokes only (possible) compromised keys rather than compromised nodes [25]. Since achieving the latter involves intrusion/misbehavior detections whose existing solutions (e.g., [26], [2]) are vulnerable to various attacks, we provide interfaces (described in Section 4.3.1) in our protocols to accommodate future solutions.

5. The threshold cryptosystem based on RSA allows a noninteractive signing protocol, which would not be the case, for example, for an ElGamal-like threshold cryptosystem.

6. The length of this period depends on the hostility of the environment that a given remote operation of the network involves and the amount of time spent for the operation; it should guarantee that no more than t dCA servers could be compromised during such a period.

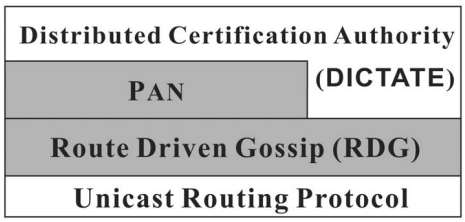


Fig. 2. The protocol structure of DICTATE: It is built upon PILOT (the gray part).

The interactions between nodes (including servers and clients) and the mCA (e.g., identification, key generation, and refreshing shared secrets) are well-defined in existing proposals [30], [27], [28], so we do not discuss them in detail; in the remainder of the paper, we rather focus on DICTATE, which maintains the online part of our certification service (i.e., the dCA). We first introduce the principles of PILOT [9], the basis of DICTATE, then we describe operations of DICTATE in detail.

4.1 Overview of PILOT

PILOT [9] is a group communication system that provides services for multicast and data replication. The parts of our PILOT system used to implement DICTATE are illustrated by the gray part in Fig. 2. PILOT is a two-layer system. It has a probabilistic multicast protocol, Route Drive Gossip (RDG), as its basis. The protocol is gossip-based in nature: It proceeds **round**[7] by round and the receivers in each round are chosen randomly (weighted according to the length of the routing path); they **relay** packets to the receivers of the later round(s) (we allow a packet to be relayed once **only** to the receivers of the **next** round in this paper). This protocol guarantees that the reliability degree, defined as the fraction of a multicast group receiving a given packet, is predictable in a probabilistic sense. The *fanout*, F, is a very important parameter related to RDG; it refers to the number of receivers chosen by a certain sender in a round and, thus, strongly influences the protocol reliability.

Upon this layer, the Probabilistic quorum system for Ad hoc Networks (PAN) provides reliable data sharing. It assumes a special group of nodes to store the shared data in a replicated manner. Any node belonging to the group is termed *server*, and the rest of the nodes are termed *clients*. A data query or update is directed to an arbitrary server, and its dissemination within the group is performed by the PAN server query protocol or update protocol (the two protocols differ in that a query requires a reply and an update does not). Since the PAN server query and update protocols rely on RDG, the probability that a query acquires the most recent update of the corresponding data object is again predictable and rather high in practice. Certain parameters, including F, have to be set when a primitive in PILOT is invoked. These parameters determine the protocol performance in terms of reliability and overhead [9].

7. The duration of a round is a parameter of RDG: It should be short enough to maintain a low propagation delay but be long enough to avoid network congestion. Although we make a synchrony assumption (in terms of rounds) in Section 5.2.2 to facilitate our analysis, it does **not** mean that PILOT is a **synchronous** system. In fact, nodes are not synchronized in our simulations.

Previous approaches for guaranteeing the reliability of queries and updates in a certification service apply either reliable group communication systems [4] or Byzantine quorum systems [10]. These approaches incur too much overhead and are thus not practical in ad hoc networks. Our approach, on one hand, builds a certification service upon probabilistic quorum systems (PAN) and, thus, provides a way of flexibly trading reliability for efficiency. On the other hand, the use of probabilistic multicast (RDG) in our approach allows us to fulfill the threshold signature scheme with $(t+1)$ anycasts.

4.2 Protocol Overview

In this section, we summarize the main operations of DICTATE. These operations are classified into external and internal (with respect to the dCA server group) protocols, according to the entities that are involved. The rationale behind these operations is explained in Section 4.3.

The notations used throughout all subsequent sections are as follows:

- s, sid: DICTATE (or dCA) server and its identity.
- c, cid: DICTATE (or dCA) client and its identity.
- $\langle m \rangle_k$: message m with signature signed by key k (identity-based signature) of a server.
- $[m]_k$: message m with signature signed by key k (RSA signature).
- K_D: public key of DICTATE (or dCA).
- k_D, k_{Ds}: private key of DICTATE (or dCA) and key share, respectively.
- $K_?$, $k_?$: public key and private key of node "?," respectively. "?" can be c, s, or a.
- $[Ct_K]_k$: certificate of public key K signed by private key k. The format of its data part is $[cid, K, v]$, where cid identifies the owner of K and v is the version number. Note that the certificate is not timestamped because it is not trivial to agree on a common timestamp among a set of servers who sign the certificate. Instead, the version number serves as a timestamp.

4.2.1 DICTATE External Protocol

This part of DICTATE runs between clients and servers.

Access Control. When a client c wants to access DICTATE, it tries to contact an arbitrary DICTATE server s, and they exchange the following messages:

$$c \rightarrow s : \mathcal{A}_c = [cid, [Ct_{K_c}]_{k_D}, rtype, seqno]_{k_c}$$
$$c \leftarrow s : \mathcal{A}_s = \langle sid, seqno \rangle_{k_s},$$

where $rtype$ refers to the request type (update or query) and $seqno$ is a sequence number. The server s admits the access of the client c with $\mathcal{A}_s$ only if $\mathcal{A}_c$ is proven to be valid. After successfully finishing this interaction, this server becomes the *agent* (later referred to as a with identity aid) of the client for this specific request.

Client Request and Server Reply. A client sends certificate update and replication requests to update its public key and related certificate, and it sends a certificate query to get the current public key certificate of another client.

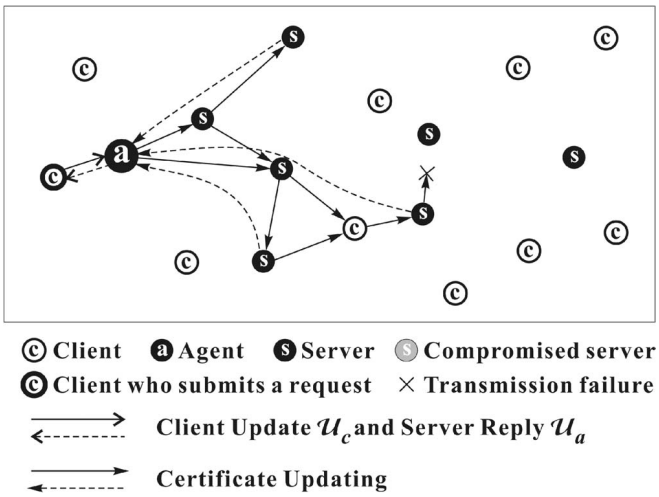


Fig. 3. DICTATE operations: Certificate Update. Note that, since DICTATE uses the service provided by a multihop routing protocol, any node (either a server or a client) can serve as relay to forward a message.

To **register a new public key**, a client c first generates a new public/private key pair K_c'/k_c' and then sends an update request to its agent a (⟶ in Fig. 3):

$$c \rightarrow a : \mathcal{U}_c = [cid, [K_c']_{k_c}]_{k_c'},$$

where k_c is the current private key whose corresponding public key K_c is to be updated. Upon completing the task of certificate update, the agent responds with the following message (⇠ in Fig. 3):

$$c \leftarrow a : \mathcal{U}_a = [Ct_{K_c'}]_{k_D}.$$

The client c verifies $[Ct_{K_c'}]_{k_D}$ using K_D. If the certificate is valid, c sends a replication request to a, so that the certificate will be stored in the dCA for future queries (⟶ in Fig. 4):

$$c \rightarrow a : \mathcal{R}_c = [cid, [Ct_{K_c'}]_{k_D}, \mathcal{F}_d]_{k_c},$$

where $\mathcal{F}_d$ indicates the required freshness degree (defined in Section 3.1). Finally, the agent a provides its client c with

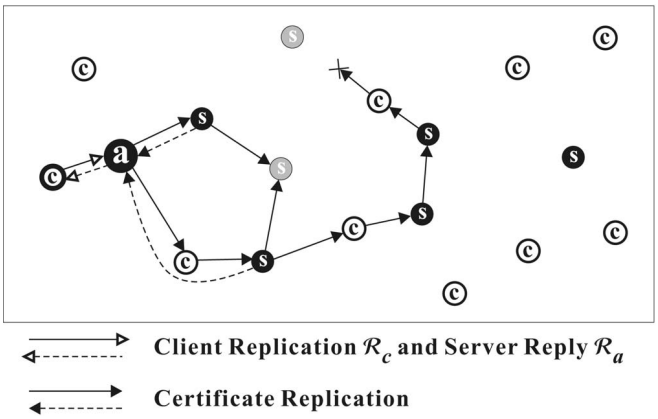


Fig. 4. DICTATE operations: Certificate Replication. Further legends can be found in Fig. 3.

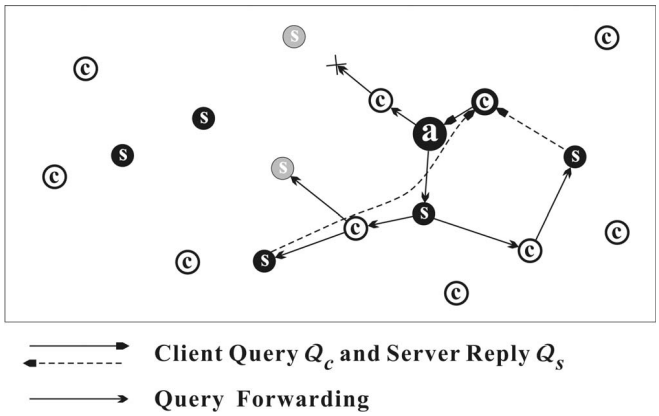


Fig. 5. DICTATE operations: Certificate Query. Further legends can be found in Fig. 3.

evidence that it has faithfully respected the protocol (⇠ in Fig. 4):

$$c \leftarrow a : \mathcal{R}_a = \langle sid_1, [Ct_{K_c'}]_{k_D} \rangle_{k_{s1}}, \langle sid_2, [Ct_{K_c'}]_{k_D} \rangle_{k_{s2}}, \cdots,$$

where $\langle sid_1, [Ct_{K_c'}]_{k_D} \rangle_{k_{s1}}, \langle sid_2, [Ct_{K_c'}]_{k_D} \rangle_{k_{s2}}, \cdots$ is a list of signatures generated by servers that receive $\mathcal{R}_c$; it proves that those servers have indeed received the new certificate. Though an abuse of concept, here we refer to $\langle sid, [Ct_{K_c'}]_{k_D} \rangle_{k_s}$ only as the signature (no plaintext included) for the concatenation of a server id sid and the certificate $[Ct_{K_c'}]_{k_D}$.

To **obtain the public key certificate of a client**, a client c sends a query request to its agent a (⟶ in Fig. 5):

$$c \rightarrow a : \mathcal{Q}_c = [cid, \hat{cid}]_{k_c}$$

where $\hat{cid}$ is the identity of the owner of the queried public key certificate. A server s that receives the request replies directly to c with the following message (⇠ in Fig. 5):

$$c \leftarrow s : \mathcal{Q}_s = \langle sid, [Ct_{K_{\hat{cid}}}]_{k_D} \rangle_{k_s}.$$

Clients involved in the aforementioned protocols should be able to perform signing (with a RSA key), signature verifications (especially pairing computations [29]), and generating public/private key pairs. While a laptop does have this capability, small mobile devices (e.g., PDA) would need more powerful processing units to perform these computations.

4.2.2 DICTATE Internal Protocol

This part of DICTATE runs among servers.

Certificate Update. An agent a forwards a valid client update to several servers via the PAN server query[8] protocol with a message $\langle aid, \mathcal{U}_c, [Ct_{K_c}]_{k_D} \rangle_{k_a}$ (⟶ in Fig. 3) and waits for enough copies of partially signed certificates $[Ct_{K_c'}]_{k_{Ds}}$ to come back (⇠ in Fig. 3). The agent then tries to combine all these certificates to create a valid one, $[Ct_{K_c'}]_{k_D}$, signed by the private key of DICTATE, and returns this new certificate back to the client via the message $\mathcal{U}_a$.

8. Although this protocol is a part of the DICTATE server update protocol, it invokes the underlying query protocol of PAN because it expects a reply from each receiver.

Certificate Replication. The agent, upon receiving a replication request $\mathcal{R}_c$ from its client, replicates the certificate via the PAN server protocols with a message $\langle aid, \mathcal{R}_c \rangle_{k_a}$ (⟶ in Fig. 4). As a consequence, a quorum Θ_U ($|\Theta_U| = 5$ in Fig. 4) of servers receives $\langle aid, \mathcal{R}_c \rangle_{k_a}$. The agent then expects acknowledgements $\{\langle sid, [Ct_{K'_c}]_{k_D} \rangle_{k_s}\} : s \in \Theta_{\bar{U}}$ from a set $\Theta_{\bar{U}}$ ($|\Theta_{\bar{U}}| = 2$ in Fig. 4) of servers (⟵ in Fig. 4), where $\Theta_{\bar{U}} \subseteq \Theta_U$. The size of Θ_U and $\Theta_{\bar{U}}$ is determined by the required freshness degree $\mathcal{F}_d$. Finally, the agent replies to the client with $\mathcal{R}_a$.

Query Forwarding. An agent a forwards a valid client query to several servers via the PAN server update protocol with a message $\langle aid, \mathcal{Q}_c \rangle_{k_a}$(⟶ in Fig. 5), but it does not expect replies from other servers. The replies $\mathcal{Q}_s$ from a quorum $\Theta_{\bar{Q}}$ ($|\Theta_{\bar{Q}}| = 2$ in Fig. 5) of servers are directly sent back to the client.

Detailed proofs of the protocol compliance with our specification are provided in Section 5 (where we also explain how to determine the quorum size). Here, we only give some intuitive ideas on the relationship between the freshness degree $\mathcal{F}_d$ and the quorum sizes:

$$\mathcal{F}_d = \Pr\{\Theta_U \cap \Theta_{\bar{Q}} \not\subseteq \mathbf{T}\}, \tag{1}$$

while $\mathcal{F}_d = 1$ iff

$$|\Theta_{\bar{U}}| + |\Theta_{\bar{Q}}| \geq n + t + 1. \tag{2}$$

Note that $\exists$ server $s : s \in \Theta_U \vee s \notin \Theta_{\bar{U}}$ could happen because a reply from s to a might not get through due to the unreliable routing protocol. It is enough that $\Theta_U \cap \Theta_{\bar{Q}} \not\subseteq \mathbf{T}$ for probabilistic guarantee (1), but a deterministic guarantee (2) requires $\Theta_{\bar{U}} \cap \Theta_{\bar{Q}} \not\subseteq \mathbf{T}$.

4.3 Protocol Rationale

This section explains the underlying principles of the DICTATE protocols.

4.3.1 DICTATE External Protocol

Access Control. A server verifies the validity of $\mathcal{A}_c$ by checking whether:

1. the nonce $\{cid, seqno\}$ is unique,
2. the certificate $[Ct_{K_c}]_{k_D}$ is valid and matches cid,
3. the signature by k_c is verifiable with K_c, and
4. the client is authorized, by the access control policy, to perform $rtype$.

We do not detail the access control policy in this paper. This policy can be specified according to different application requirements and, in particular, provides an interface to accept inputs from certain intrusion/misbehavior detection algorithms (whose accusations against certain nodes disallow the access of those nodes and hence revoke their certificates). The access control phase is important because it allows a request (especially for certificate update) to be checked against certain policies before the client actually performs the computationally intensive key generation. The server maintains a state for each valid request until the request is fulfilled.

A client updates the nonce if it needs to retransmit $\mathcal{A}_c$ (e.g., due to the loss of $\mathcal{A}_s$). A server replies to every $\mathcal{A}_c$ that has a unique nonce (but only keeps one request state for the client if the time interval between two requests is too short). As a result, no reply $\mathcal{A}_s$ would be sent to an adversary that launches a replay attack.

Client Request and Corresponding Reply. The role of the update request $\mathcal{U}_c$ is to prove to the agent that the client c owns the private keys k_c and k'_c corresponding to the current public key K_c and the newly proposed public key K'_c, respectively. The agent is able to check the validity of $\mathcal{U}_c$ given the $[Ct_{K_c}]_{k_D}$ transferred in the access control phase.

If an agent were trustworthy, it could be asked to directly perform the replication after obtaining $[Ct_{K'_c}]_{k_D}$, without notifying its client. However, we want to protect the client against possible compromise of the agent. The server reply $\mathcal{U}_a$ allows the client to check the validity of the new certificate before asking the agent to replicate it. Also, each server s that receives the replication request $\mathcal{R}_c$ is required to provide the "receipt" $\langle sid, [Ct_{K'_c}]_{k_D} \rangle_{k_s}$ to the client. This scheme defends DICTATE against a compromised agent who pretends to be correct by finishing the certificate update task but fails to replicate the certificate to other DICTATE servers afterward. In addition, the client mentions its required freshness degree[9] $\mathcal{F}_d$, so that the agent can set proper parameters to invoke the underlying protocols according to the requested degree. There is no way for both a replication and a query to request a freshness degree, otherwise the system performance becomes unpredictable. Although greedy clients might always request the highest freshness degree (i.e., $\mathcal{F}_d = 1$), which incurs a large protocol overhead, DICTATE discourages greed by means of an implicit "self-castigation:" The higher $\mathcal{F}_d$, the longer the delay of the response. Note that a lower freshness degree does **not** mean a lower level of security; we refer to Section 5.2.2 for detailed explanations.

A query request $\mathcal{Q}_c$ is propagated through the agent, as it needs to go through certain sanity checks. But, the reply should not go through the agent; otherwise, a compromised agent could return bogus information to the client and the client has no way of knowing whether the agent is telling the truth or not. Therefore, the client should collect all replies and select the most recent version of the required certificate by itself.

4.3.2 DICTATE Internal Protocol

In previous literature (e.g., [10]), it is considered very important that a CA be independent from other network nodes. This requirement improves the scalability of a network because, for example, a client does not need to be informed about the revocation of a server public key. DICTATE is also divided into two parts, i.e., external and internal, for the same purpose. However, since we do not want to trust an agent (the bridge between the two parts), we would like a client to be able to verify the response from any server. This explains why we apply an identity-based public-key system for DICTATE servers: It grants the ability of verifying servers' responses to a client without jeopardizing the network scalability. Alternatively, a solution to our problem can be based on the concept of *Byzantine Fault Tolerance* [10]. Such a solution would let a client access a set

9. This value is set by an application without involving a human user, based on the value of the communication that will use the key later.

of agents ($t+1$ in the case of t possible compromised servers) to "mask" the failure. Unfortunately, a resulting scheme would leave the certification service open to easily launched resource-clogging DoS attacks in wireless networks and cause a well-behaved client to experience a very long delay for a request.

No underlying secure group communication scheme (e.g., [31]) is needed to support DICTATE. The system is built upon PILOT, which is in turn based directly on unicast routing; therefore, it can secure itself with the prerequisite that the public key of each server is known to the whole network.

Certificate Update. An agent a signs the update request $\mathcal{U}_c$ and the previous certificate $[Ct_{K_c}]_{k_D}$ from its client c and disseminates this message within DICTATE via the PAN server query protocol.[10] Each server again verifies the validity of $\mathcal{U}_c$ in the same way the agent did. A server s, upon validating the request from a, updates the certificate Ct_{K_c} by increasing the version number v as $v = v+1$ and substituting K_c with K'_c. Then, it generates a partially signed certificate $[Ct_{K'_c}]_{k_{D_s}}$ and sends it to a.

Determining the parameters (including the fanout F, see Section 4.1) to invoke PAN is a key issue to ensuring a successful completion of the threshold signing procedure because an agent should collect at least $t+1$ partially signed certificates to obtain a valid certificate signed by the private key of DICTATE. According to the analysis provided in Section 5.2.2, the agent can decide on values of certain parameters by which the PAN server query protocol reaches more than $t+1$ servers. The agent might need to invoke this procedure more than once before finishing.

The other issue is how to transfer the responses to a query (with respect to PAN) back to an agent. This problem is common to all internal protocols. In [9], we suggest a direct call to the unicast routing primitive. The strengths of this approach are: 1) the complexity of the protocols is low and 2) a compromised server cannot block a transmission. Unfortunately, several parallel transmissions to a common destination may congest the network. *Reverse path forwarding* is an alternative. As illustrated in Fig. 3, a reply follows the tree created in the dissemination phase back to the agent and there are packet-level aggregations at internal vertices of the tree. This approach improves the protocol efficiency if the percentage of compromised servers is low, but suffers from a high protocol complexity. Even worse, a compromised server s launching a DoS attack by blocking the transmission in the reply phase can greatly reduce the efficiency of the protocol, especially when s is close to the agent. We suggest that an agent switches between unicast routing and reverse path forwarding based on the estimation of the percentage of compromised servers.

Certificate Replication. This protocol is built upon both the PAN server update protocol and the PAN server query protocol. The parameters for invoking PAN are set to meet the freshness requirement $\mathcal{F}_d$ of the client. How to estimate the size of the resulting quorum Θ_U under certain parameters is described in Section 5.2.2. Each server, upon receiving the message $\langle aid, \mathcal{R}_c \rangle_{k_a}$, verifies its validity and then stores the certificate $[Ct_{K'_c}]_{k_D}$. If the client requests the highest freshness degree, the PAN server query protocol is invoked to require a reply $\langle sid, [Ct_{K'_c}]_{k_D} \rangle_{k_s}$ from each server $s \in \Theta_U$, and the resulting reply quorum $\Theta_{\bar{U}}$ should respect (2); otherwise, the PAN server update protocol is invoked. In the latter case, replies are only required from those servers receiving a replication request directly from the agent, which is already enough to prove the agent's compliance with the protocol; the resulting freshness degree is estimated by (1).

Query Forwarding. The agent's task is trivial in this protocol. It simply forwards a valid query request to other servers via the PAN server update protocol. Each server receiving the query request replies directly to the client with its own copy of the certificate, if the query is valid (proved by checking the signature of the agent). The client waits for a certain period of time until it collects the replies from a quorum $\Theta_{\bar{Q}}$ of servers. The minimum size expected for $\Theta_{\bar{Q}}$ is $|\Theta_{\bar{Q}}|_{min}$, which is set by DICTATE (based on the network size and the query rate [9]) and known to all nodes; it cannot be modified by a client (in contrast to the situation of $|\Theta_{\bar{U}}|$ whose value can be modified by a client through $\mathcal{F}_d$). Within this collection, the certificate with the highest version number is chosen as the response to this query.

5 Security Analysis

In this section, we verify the security of DICTATE against the properties specified in Section 3.1 by considering malicious/compromised clients and compromised servers separately.

5.1 Protection from Malicious/Compromised Clients

Malicious clients have no way to impersonate other nodes, because a server can verify the identity of a client by checking the ownership of the private key corresponding to a certified public key. Hence, a malicious client cannot impair the safety property.[11] If the certificate update is performed frequently, an attacker is given virtually no chance to compromise a private key before the key expires. All exchanged messages are authenticated (signed by their senders) but do not require confidentiality, so attacks such as eavesdropping and message corruption cannot degrade the security of DICTATE. However, DICTATE can suffer from DoS attacks launched by clients replaying certain requests to cram the service queue. Defending against such attacks is difficult because a server should anyway verify a request before knowing its legitimacy. Actually, DICTATE already decreases the risk of such a DoS attack by involving only one agent for each request. Another kind of DoS attack is routing disruption [12]. Fortunately, the liveness property of DICTATE can be guaranteed, provided that, between a

10. This protocol will be used for several purposes in DICTATE. Such flexibility is granted by a callback procedure [9] included in PAN, which can be defined to retrieve any information (e.g., $[Ct_{K'_c}]_{k_{D_s}}$) from the upper layer (DICTATE in this case).

11. A bogus certificate update could be disseminated within DICTATE if this update request were admitted by a compromised agent. Since this attack has the same effect as an agent propagating a fictitious update or replica, we refer to Section 5.2 for a detailed analysis.

sender and a receiver, there exists at least one routing path that contains no malicious nodes.

Compromised clients, when behaving as described above, cannot compromise DICTATE. However, owning the identities and certificates of once legitimate nodes, they pose potential threats to other nodes that still believe in their legitimacy. DICTATE does not directly thwart these threats; it rather relies on inputs from misbehavior detection algorithms (e.g., reputation systems [32], [33]) to identify and thus evicts these nodes (we refer to Section 4.3.1 for details).

5.2 Defense against Compromised Servers

The attacks performed by a given compromised server vary with the role of that server.

5.2.1 Compromised Agents

A compromised agent may decline to serve a request for a client. A client can detect such an attack by setting a timer for each request and can change the agent should the timer expire. If an agent fulfilled the task of certificate update but tried to cheat its client without following up the replication, it would fail to provide evidences

$$\langle sid_1, [Ct_{K_c'}]_{k_D}\rangle_{k_{s1}}, \langle sid_2, [Ct_{K_c'}]_{k_D}\rangle_{k_{s2}}, \cdots.$$

In both cases, the liveness property of DICTATE is ensured, provided that the client eventually finds some correct server as its agent. A compromised agent may also provide fictitious requests (or replies) to other servers (or its clients). Such behavior does not compromise the safety property of DICTATE because a message receiver can always verify the validity of the message signed by its original sender. Although fictitious requests disseminated by a compromised agent do reduce the service capacity of DICTATE, this kind of DoS attack can be easily detected, so that correct servers may convoke an emergency checktime in order to purge the compromised server.

5.2.2 Compromised Common Servers

We call a *common server* a server that does not have the role of the agent for a given request. A compromised common server can 1) issue partially signed certificates of any bitstring, 2) behave as a malicious node, and 3) deviate from the protocol requirements by omitting the verification of all replicas (which leads to a later reply to a query with an obsolete certificate) or by simply not storing or forwarding a replica. DICTATE defends itself against 1) by using a $(t+1, n)$ threshold cryptosystem, such that the liveness and safety properties are guaranteed as long as the total number of compromised servers is no more than t. For the second type of attacks, we refer to Section 5.1 for related discussions. The third type consists in attacks of our particular interests; we consider them to be DoS attacks that we term *inaction* DoS (or iDoS) and analyze them in detail.

Since DICTATE is built upon a probabilistic quorum system, the degree $\mathcal{F}_d$ that iDoS attacks will not compromise the freshness property of DICTATE is the probability that $\exists s : s \in \mathbf{C} \cap \Theta_U \cap \Theta_{\bar{Q}}$ (i.e., there exists at least one correct server that receives both a replica and a later query to the replica), as expressed in (1). In this sense, DICTATE can be considered as a special instance of (b, ε) dissemination quorum systems [23] ($b = t$ and $\varepsilon = 1 - \mathcal{F}_d$ in our case), innovating on the system design with an asymmetric quorum construction and a randomized quorum size [9]. To compute $\mathcal{F}_d$, we rewrite (1) by using the concept of combination and also by taking expectations over probability distributions (of corresponding random variables) as follows:

$$\begin{aligned}
\mathcal{F}_d &\geq \sum_{r=0}^{\bar{r}} \sum_{i=0}^{n} \left(1 - \frac{A}{B}\right) \nu_r(i) p_r \\
A &= \binom{n - |\Theta_U^r|}{|\Theta_{\bar{Q}}|_{min}} \text{ is the number of events where } \Theta_U \cap \Theta_{\bar{Q}} = \emptyset, \\
B &= \binom{n}{|\Theta_{\bar{Q}}|_{min}} \text{ is the total number of events of taking } |\Theta_{\bar{Q}}|_{min} \text{ out of } n,
\end{aligned} \tag{3}$$

where ν_r is the probability distribution of $|\Theta_U^r|$ (i.e., $\nu_r(i) = \Pr\{|\Theta_U^r| = i\}$), and p_r is the probability that a query occurs $r+1$ rounds[12] later than when the considered certificate was replicated. Note that both $|\Theta_U^r|$ and ν_r are functions of r. In order to simplify the case, we compute only the lower bound of $\mathcal{F}_d$ by plugging in the minimum value of $|\Theta_{\bar{Q}}|$ set by DICTATE. Now, we show how to calculate ν_r with a recurrence relation defined by a Markov chain. Note that the Markov chain is 2D because the number of servers that will receive a message in the next round depends on the **increase** in the number of servers that have received the message in the current round (see Section 4.1 for details). Let $S_r = |\Theta_U^r|$ and $\mathbf{S}_r$ be a vector $[S_r, S_{r-1}]_{r\geq 0}^T$ for brevity; the distribution of S_r is estimated with the following recurrence relation, with the initial condition $\Pr\{\mathbf{S}_0 = [1, 0]^T\} = 1$:

$$\begin{aligned}
&\Pr\left\{\mathbf{S}_{r+1} = \begin{bmatrix} j \\ i \end{bmatrix}\right\} \\
&= \sum_{i_1=0}^{i} \binom{n-i}{j-i} (1 - q^{i-i_1})^{j-i} q^{(i-i_1)(n-j)} \Pr\left\{\mathbf{S}_r = \begin{bmatrix} i \\ i_1 \end{bmatrix}\right\}, \\
&\nu_r(i) = \sum_{i_1=0}^{i} \Pr\left\{\mathbf{S}_r = \begin{bmatrix} i \\ i_1 \end{bmatrix}\right\},
\end{aligned} \tag{4}$$

where

$$q = 1 - \left(\frac{F}{n-1}\right)\left(\frac{n-t}{n-1}\right)\mathbf{E}_H[(1-p_f)^H] \tag{5}$$

is the probability that a certain server does not receive the propagated message in a given gossip round (recall that F is the fanout, see Section 4.1). This expression takes into account the probability that either 1) the server is not chosen as a gossip destination, 2) the server is compromised (we assume the worst case iDoS attack where a compromised server drops any message it receives), or 3) the message fails to reach its destination: as p_f is an identical and independent probability of failure for each node along a routing path in a certain

12. We assume that nodes gossip in synchronous rounds for the analysis, but our protocols do not rely on synchronization in practice.

network environment and H is a random variable representing the length of an arbitrarily chosen routing path, the expectation (or $\mathbf{E}_H$) of $(1-p_f)^H$ characterizes the end-to-end failure probability. We refer to [9] for detailed discussions.

DICTATE is not limited to the probabilistic semantics of the freshness. A deterministic assurance can be provided by meeting the requirement of (2) such that $|\Theta_{\bar{U}} \cap \Theta_{\bar{Q}}| \geq t+1$ and, thus, $\Theta_{\bar{U}} \cap \Theta_{\bar{Q}}$ includes at least one correct server. In this case, DICTATE becomes an instance of strict dissemination quorum systems [24]. Of course, such a protocol setting incurs much higher overhead than that of a probabilistic protocol. As we mentioned in Section 4.3.1, a lower freshness degree does not mean a lower level of security. Since DICTATE provides an online revocation service, the goal of a client that queries the status of a given certificate is to check the validity of the corresponding public key obtained from the other client. Although there is a rare chance of obtaining an incorrect status due to the probabilistic nature of DICTATE, it does not compromise the security of the revocation service but only reduces its efficiency (because an incorrect status prevents the client from using a correct public key before the client completing a successful query).

6 Simulations

In this section, we verify the performance of DICTATE with respect to the freshness degree $\mathcal{F}_d$, under the iDoS attacks launched by compromised servers.

6.1 Parameters and Assumptions

We use *ns-2* [34] with the Monarch Project wireless and mobile extensions. This simulator provides both implementations of ad hoc routing protocols (we use DSR as an example) and wireless MAC, based on the Lucent WaveLAN IEEE 802.11 product, with a 2Mbps transmission rate and a nominal range of 250m.

We simulate an ad hoc network with 100 nodes in a square area of 1km^2. The movement pattern is defined by the "random waypoint" model [35], and we update it by setting a positive minimum speed (as suggested by [36]). We pair the mobility parameters, such that each node has a maximum speed of 2m/s, 5m/s, 10m/s, and 20m/s, and a corresponding average pause time of 10s, 20s, 40s, and 80s, respectively.

In our simulations, the dCA contains 19 servers, which means that the system can sustain up to $6 = (19-1)/3$ compromised servers. The dCA servers are assumed to be predefined. The external protocols and the internal certificate update protocol of our DICTATE are omitted to simplify the interpretation of the results. Certificate replications and queries are assumed to be independent Poisson arrival processes with intensities λ_R and λ_Q, respectively. They are emulated by Poisson traffic sources attached to each server, generating packets of 512 bytes.[13] The overall access rate $\lambda_O = \lambda_R + \lambda_Q$ is set at $2s^{-1}$, and we also assume that $\lambda_O = 8\lambda_R$. This allows each client to update its certificate about every 5 minutes and to query certificates of other clients every 45 seconds, which is more than enough to thwart key compromises and impersonations. The duration of a gossip round is set to 200ms, and the value of $|\Theta_{\bar{Q}}|_{min}$ is set to 3 for all simulations.[14]

We assume that all replication and query requests are targeted at the same certificate. The assumption might seem to be exaggeratedly pessimistic because the chance that a queried certificate has just been updated is negligible (recall that there are in total $100-19=81$ certificates to be updated and queried). However, the goal is to force a query to always return the result of the latest replication and thus to illustrate the lower bound of the freshness degree that is provided by DICTATE. We term the resulting freshness degree *pessimistic* $\mathcal{F}_d$. In addition, we also evaluate *optimistic* $\mathcal{F}_d$: A query is considered to be successful even if it only returns the result of the penultimate replication.

We first investigate the impact of t (the number of compromised servers) on the performance of DICTATE, then we show how the analytical results in Section 5.2.2 can be used to tune the freshness degree to a required value. DICTATE is operated over 400 seconds of simulated time. The first 30 seconds of the simulation are used for system initialization. Then, each traffic source continues generating traffic according to the predefined intensity until the end. Each simulation was carried out 10 times with different scenario files created by *ns-2*.

6.2 Impact of t on DICTATE Performance

We first set the fanout used for certificate replication at $F=2$ and vary the number of compromised servers t, as well as node mobility parameters. Note that we always designate the mobility pair by the maximum node speed. As shown in Figs. 6a and 6b, the freshness degree $\mathcal{F}_d$ degrades modestly even in the worst case (i.e., $t=6$) for low mobility scenarios (i.e., $Speed_{max} = 2 \text{ or } 5\text{m/s}$). In these cases, we can claim that $\mathcal{F}_d \geq 0.95$ in practice, considering that $t=6$ is a rare case and the pessimistic $\mathcal{F}_d$ shows the lower bound of the DICTATE performance. However, the effect of t is significant for high mobility scenarios (i.e., $Speed_{max} = 10 \text{ or } 20\text{m/s}$). In both cases where $t = 4 \text{ and } 6$, the pessimistic $\mathcal{F}_d$ drops to values around 0.80. The system has to adjust itself if, in such situations, a freshness degree higher than 0.90 is requested by a client. We show the tunability of DICTATE in Section 6.3.

The network load incurred by DICTATE is shown in Fig. 6c. This load is reasonably low since each request costs less than eight unicastings on average, given that there are two requests per second and the expected length of a routing path is about two hops (for our simulation scenarios). If a traditional access protocol were used for these requests, which would mean accessing DICTATE servers by multiple unicastings, the resulting load would be much higher. It can also be observed that the larger the number of compromised servers, the lower the load (because a compromised server drops any message instead of relaying it).

13. This is approximately the size of $\langle aid, \mathcal{R}_c \rangle_{k_a}$, the largest message involved in the simulated part of DICTATE, if the RSA key is 1,024 bits and the identity-based ECC key is 163 bits.

14. Although DICTATE may not always guarantee $\mathcal{F}_d = 1$ with $|\Theta_{\bar{Q}}|_{min} = 3$ when $t > 1$, the simulation results show that $\mathcal{F}_d$ can be made as close to 1 as possible even when $t > 1$.

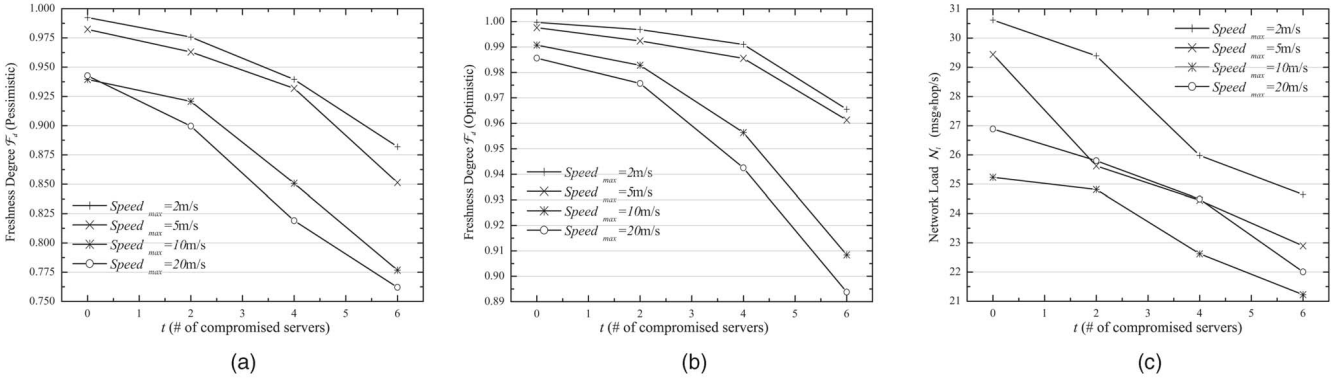


Fig. 6. Freshness degree $\mathcal{F}_d$ and network load versus the number of compromised servers t, under four mobility scenarios. The fanout for certificate replication is set at $F = 2$. (a) Pessimistic freshness degree. (b) Optimistic freshness degree. (c) Network load.

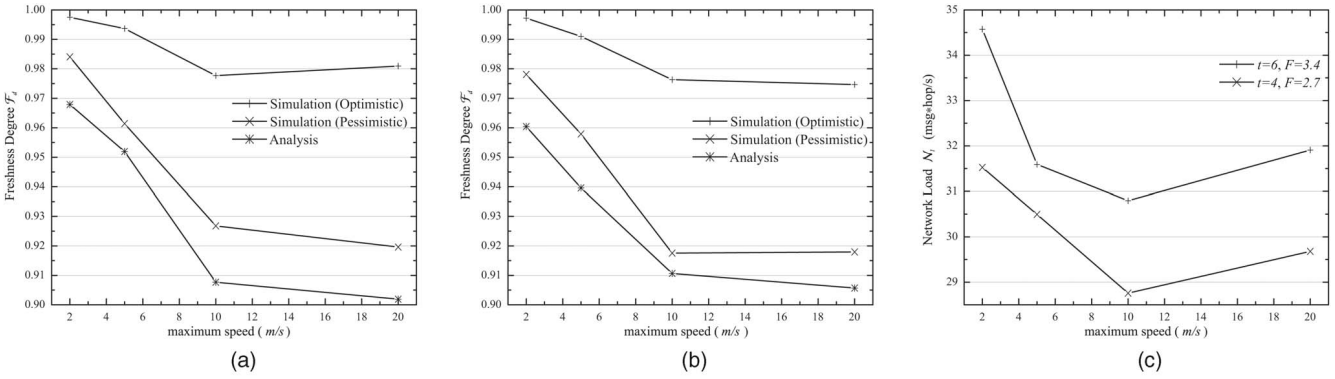


Fig. 7. Freshness degree $\mathcal{F}_d$ and network load versus the node mobility. The fanout for certificate replication is adjusted according to analytical results, in order to cope with a large proportion of compromised servers and high node speed. (a) Fresheness degree with $t = 4,\ F = 2.7$. (b) Fresheness degree with $t = 6,\ F = 3.4$. (c) Network load.

6.3 Tuning the Freshness Degree

As we have already mentioned, the parameter settings used in Section 6.2 are not appropriate to cope with a large proportion of compromised servers in high mobility scenarios. So, if an agent is aware of a significant increase in compromised servers, it will adjust the parameter settings for the internal protocols. The simulations described in this subsection illustrate how such adjustments are performed based on the analysis in Section 5.2.2, given a required freshness degree of 0.90.

A fanout F that ensures $\mathcal{F}_d \geq 0.90$ can be deduced from expressions (3) to (5), given a particular value of t and certain network conditions. The analytical results for $t = 4 \text{ and } 6$ are shown by the bottom curves in Figs. 7a and 7b, respectively. The simulation results also provided in these two figures further prove that the fanouts resulting from the analysis ($F = 2.7$ for $t = 4$ and $F = 3.4$ for $t = 6$)[15] indeed lead to satisfactory freshness degrees, i.e., the experimental values of $\mathcal{F}_d$ are always higher than the predicted values. Of course, the system actually trades its overhead for the freshness degree in these cases. The comparison between Fig. 6 and Fig. 7 shows that an increase of network load up to 50 percent is traded for an improvement of about 300 percent of the freshness degree (i.e., $1 - \mathcal{F}_d$ is divided by 3) in the extreme situation: $t = 6$ and $Speed_{max} = 20\text{m/s}$.

15. A real number $x.y$ for F means that each server, when propagating a message, takes $F = x$ with probability $1 - y/10$ and $F = x + 1$ with probability $y/10$.

7 Conclusion

In this paper, we have focused on the design of a certification authority in ad hoc networks. We take a joint authority approach that combines an offline identification authority and an online distributed revocation authority. We have then proposed DICTATE, based on our previous work on reliable group communication systems, to control the online authority. The originality of DICTATE includes 1) flexible certificate management protocols with tunable freshness to trade overhead for robustness, and 2) provable robustness against various attacks, especially Byzantine failures of the DICTATE servers.

Our proposed specification of distributed CA with probabilistic freshness takes the peculiarities of ad hoc networks into account. As a consequence, the freshness property can be sacrificed to some extent, in the case that the required freshness degree is low and the network resources are scarce. In order to meet the specification, DICTATE is implemented in a "closed" server group backed by an identity-based cryptosystem, so that the Sybil attack is thwarted and messages from individual servers are universally verifiable. Also, the authority employs threshold cryptography to distribute trust in order to tolerate a certain number of compromised servers. Finally, DICTATE relies on a probabilistic group communication system, PILOT, to propagate certificate updates, replications, and queries to its server group, which guarantees a certain probability, rather high in practice, for a query to obtain the

latest status of a certificate. Despite the seeming complexity, DICTATE executes automatically without the need of any human involvement.

We have identified potential attacks against DICTATE and evaluated the robustness of DICTATE through detailed security analysis. We notice that one of these attacks, called inaction DoS (iDoS) attack, is a serious threat to DICTATE; we have thus verified the system performance under such attacks by simulations. The results show that 1) iDoS attacks can only have a marginal effect on the performance of DICTATE in low node speed scenarios and 2) increasing node speed weakens the tolerance of DICTATE to such attacks. In the latter case, the system can be tuned online to trade its overhead for a higher degree of freshness. We have improved the analytical model proposed for PILOT to predict the freshness degree of DICTATE. The validity of predictions is evaluated by simulations in a way that the analytical results are used as the basis to adjust system parameters for tuning the DICTATE performance.

We are in the process of further studying the performance of DICTATE with a complete implementation. It is also a part of our future work to consider the integration of DICTATE with potential applications such as secure routing and secure group communications.

ACKNOWLEDGMENTS

The authors would like to thank Markus Jakobsson, Gildas Avoine, Naouel Ben Salem, and Mario Cagalj for several instructive discussions. The authors are also grateful to the editor and the anonymous referees for their insightful comments that helped improve the quality of this paper. This work was supported (in part) by National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322 (http://www.mics.org).

REFERENCES

[1] L. Zhou and Z. Haas, "Securing Ad Hoc Networks," *IEEE Network,* vol. 13, no. 6, pp. 24-30, 1999.

[2] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang, "URSA: Ubiquitous and Robust Access Control for Mobile Ad-Hoc Networks," *IEEE/ACM Trans. Networking,* vol. 12, no. 6, 2004.

[3] *Public-Key Infrastructure (X. 509),* PKIX Working Group, The Internet Eng. Task Force (IETF), http://www.ietf.org/html.charters/pkix-charter.html, 2005.

[4] M.K. Reiter, M.K. Franklin, J.B. Lacy, and R.N. Wright, "The Ω Key Management Service," *J. Computer Security,* vol. 4, no. 4, pp. 267-287, 1996.

[5] P. Zimmermann, *The Official PGP User's Guide.* MIT Press, 1995.

[6] S. Čapkun, L. Buttyán, and J.-P. Hubaux, "Self-Organized Public-Key Management for Mobile Ad Hoc Networks," *IEEE Trans. Mobile Computing,* vol. 2, no. 1, pp. 52-64, 2003.

[7] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in Distributed Systems: Theory and Practice," *ACM Trans. Computer Systems,* vol. 10, no. 4, pp. 265-310, 1992.

[8] S. Stubblebine, "Recent-Secure Authentication: Enforcing Revocation in Distributed System," *Proc. IEEE Conf. Security and Privacy,* 1995.

[9] J. Luo, P.Th. Eugster, and J.-P. Hubaux, "PILOT: ProbabilistIc Lightweight grOup communication sysTem for Mobile Ad Hoc Networks," *IEEE Trans. Mobile Computing,* vol. 3, no. 2, pp. 164-179, 2004.

[10] L. Zhou, F.B. Schneider, and R. van Renesse, "COCA: A Secure Distributed Online Certification Authority," *ACM Trans. Computer Systems,* vol. 20, no. 4, pp. 329-368, 2002.

[11] *Simple Public Key Infrastructure (SPKI),* SPKI Working Group, The Internet Eng. Task Force (IETF), http://www.ietf.org/html.charters/spki-charter.html, 2004.

[12] Y. Hu, A. Perrig, and D.B. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Proc. ACM MobiCom'02 Conf.,* 2002.

[13] M.G. Zapata and N. Asokan, "Securing Ad Hoc Routing Protocols," *Proc. ACM WiSe'02 Conf.,* 2002.

[14] Y. Hu, A. Perrig, and D.B. Johnson, "Packet Leashes: A Defense Against Wormhole Attacks in Wireless Networks," *Proc. IEEE INFOCOM'03 Conf.,* 2003.

[15] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang, "Self-Securing Ad Hoc Wireless Networks," *Proc. IEEE Int'l Symp. Computers and Comm.,* 2002.

[16] J. Douceur, "The Sybil Attack," *Proc. of Int'l Workshop Peer-to-Peer Systems,* 2002.

[17] J.-P. Hubaux, L. Buttyán, and S. Čapkun, "The Quest for Security in Mobile Ad Hoc Networks," *Proc. Mobile Ad Hoc Networking and Computing Workshop,* 2001.

[18] S. Yi and R. Kravets, "MOCA: Mobile Certificate Authority for Wireless Ad Hoc Networks," *Proc. Ann. PKI Research Workshop Program,* 2003.

[19] M. Bechler, H.-J. Hof, D. Kraft, F. Pahlke, and L. Wolf, "A Cluster-Based Security Architecture for Ad Hoc Networks," *Proc. IEEE INFOCOM '04 Conf.,* 2004.

[20] A. Khalili, J. Katz, and W.A. Arbaugh, "Toward Secure Key Distribution in Truly Ad-Hoc Networks," *Proc. IEEE Workshop Security and Assurance in Ad hoc Networks,* 2003.

[21] A. Weimerskirch and G. Thonet, "A Distributed Light-Weight Authentication Model for Ad-Hoc Networks," *Proc. Fourth Int'l Conf. Information Security and Cryptology (ICISC 2001),* 2001.

[22] G. Montenegro and C. Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Adresses," *Proc. Network and Distributed System Security Symp.,* 2002.

[23] D. Malkhi, M.K. Reiter, A. Wool, and R.N. Wright, "Probabilistic Quorum Systems," *Information and Computation,* vol. 170, no. 2, pp. 184-206, 2001.

[24] D. Malkhi and M.K. Reiter, "Byzantine Quorum Systems," *Distributed Computing,* vol. 11, no. 4, pp. 203-213, 1998.

[25] W. Stallings, *Cryptpgraphy and Network Security: Principle and Practices.* Prentice Hall, 2003.

[26] Y. Zhang, W. Lee, and Y. Huang, "Intrusion Detection Techniques for Mobile Wireless Networks," *ACM/Kluwer Wireless Networks,* vol. 9, no. 5, pp. 545-556, 2003.

[27] R. Gennaro, T. Rabin, S. Jarecki, and H. Krawczyk, "Robust and Efficient Sharing of RSA Functions," *J. Cryptology,* vol. 13, no. 2, pp. 273-300, 2000.

[28] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *Proc. Ann. Int'l Cryptology Conf.,* 2001.

[29] J.C. Cha and J.H. Cheon, "An Identity-Based Signature from Gap Diffie-Hellman Groups," *Proc. Sixth Int'l Workshop Theory and Practice in Public Key Cryptography: Public Key Cryptography,* 2003.

[30] A. Menezes, P. van, Oorschot, and S. Vanstone *Handbook of Applied Cryptography.* CRC Press, 1997.

[31] M. Steiner, G. Tsudik, and M. Waidner, "Key Agreement in Dynamic Peer Groups," *IEEE Trans. Parallel and Distributed Systems,* vol. 11, no. 8, pp. 769-780, 2000.

[32] S. Buchegger and J.-Y. Le Boudec, "Performance Analysis of the CONFIDANT Protocol (Cooperation of Nodes—Fairness in Dynamic Ad-Hoc NeTworks)," *Proc. ACM Int'l Symp. Mobile Ad-Hoc Networking and Computing,* 2002.

[33] C. Crepeau and C.R. Davis, "A Certificate Revocation Scheme for Wireless Ad Hoc Networks," *Proc. of ACM Workshop Security of Ad Hoc and Sensor Networks,* 2003.

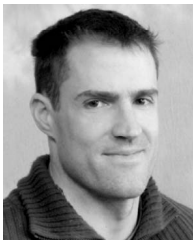
[34] *The ns Manual,* K. Fall and K. Varadhan, eds., The VINT Project, Univ. of California at Berkeley, LBL, USC/ISI, and Xerox PARC, Apr. 2002, http://www.isi.edu/nsnam/ns/.

[35] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing,* T. Imielinski and H. korth, eds., pp. 153-181, 1996.

[36] J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," *Proc. of IEEE INFOCOM'03 Conf.,* 2003.

Jun Luo received the BS and MS degrees both in electrical engineering from Tsinghua University, Beijing, PRC, in 1997 and 2000, respectively. As a research assistant of Laboratory for Computer Communication and Application (LCA), he is now working toward the PhD degree in computer science in EPFL (Swiss Federal Institute of Technology in Lausanne). His research interests include mobile computing (especially in ad hoc and sensor networks), reliable group communication, and network security. He is a student member of the ACM and IEEE.

Jean-Pierre Hubaux joined the faculty of the Swiss Federal Institute of Technology Lausanne (EPFL) in 1990; he was promoted to full professor in 1996. His research activity is focused on mobile networking and computing, with a special interest in fully self-organized wireless ad hoc networks. In particular, he has performed research on cooperation aspects, security, power efficiency, and distributed algorithms for ad hoc and sensor networks. During the last few years, he has been strongly involved in the definition and launching phases of a new National Competence Center in Research named Mobile Information and Communication Systems (NCCR/MICS), see http://www.terminodes. org. He served as the general chair for the Third ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002), held in June on the EPFL campus. He is an associate editor of *IEEE Transactions on Mobile Computing* and of the *Elsevier Journal on Ad Hoc Networks*. Within his first year at EPFL, he defined the first curriculum in communication systems. From October 1999 until September 2001, he was the first chairman of the Communication Systems Department. He has held visiting positions at the IBM T.J. Watson Research Center and at the University of California at Berkeley. He has published more than 60 papers in the area of networking. Formerly, he spent 10 years in France with Alcatel, where he was involved in R&D activities, mostly in the area of switching systems architecture and software. He is a senior member of the IEEE.



Patrick T. Eugster received the PhD and MS degree in computer science both from the Swiss Federal Institute of Technology in Lausanne, Switzerland (EPFL). He is an assistant professor at Purdue University. His research focuses on abstractions, languages, and algorithms for distributed programming.